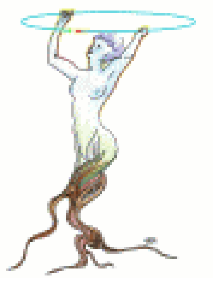


What Else is New Since ROOT 2000?

Fons Rademakers

New Build System

- Move from CMZ to well known Open Source tools like CVS and (g)make
- To build ROOT on any platform do:
  - ./configure <platform>; make; make install
- We don't use autoconf and automake since most platform ifdef's were already in the source, and we already had figured out how to build shared libs on all platforms, but the idea is the same

Makefile Structure

- The ROOT Makefile has been structured as described in the paper: "Recursive Make Considered Harmful“
  - **http://www.tip.net.au/~millerp/rmch/recu-make-cons-harm.html**
- The main philosophy is that it is better to have a single large Makefile describing the entire project than many small Makefiles, one for each sub-project, that are recursively called from the main Makefile. By cleverly using the include mechanism the single Makefile solution is as modular as the recursive approach without the problems of incomplete dependency graphs.

Makefile Features

- The single Makefile is FAST
  - about 1 sec to check if anything needs to be recompiled
- The Makefile supports parallel builds
  - make –j 24 on FermiLab’s SGI’s
- The ROOTBUILD shell variable can be used to steer build
  - export ROOTBUILD=debug

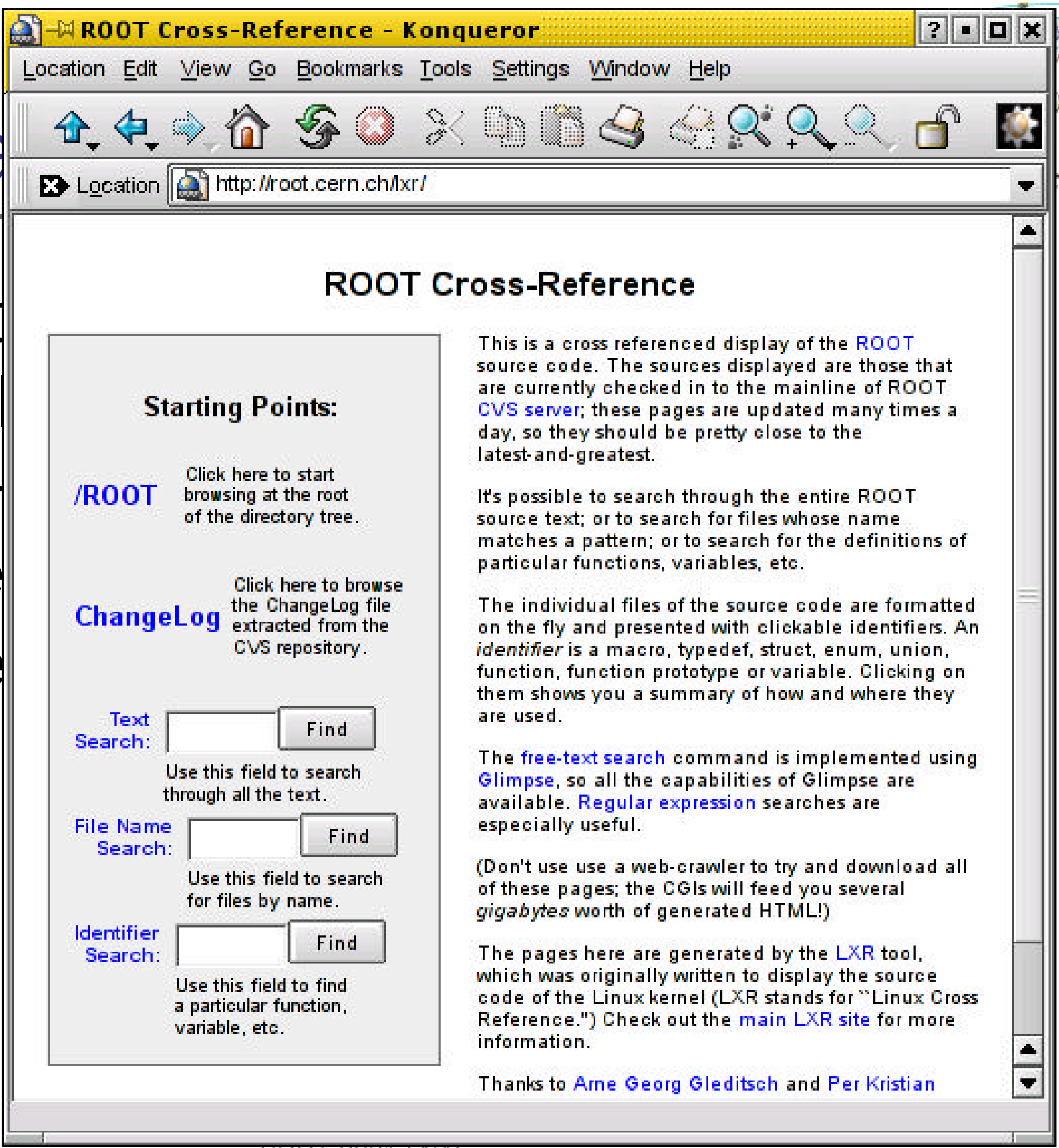
Important Makefile Targets

make all	(default)
make install	(install to path specified in ./configure)
make dist	(binary tar.gz distribution)
make redhat	(build binary rpm, by Christian Holm)
make debian	(build binary pkg, by Christian Holm)
make distsrc	(source tar.gz)
make distclean	(clean everything except configure info)
make maintainer-clean	(distclean + remove configure info)
make cintdlls	(build all CINT add-on dll's)
make html	(generate HTML documentation of classes)

make all-<module>	(builds everything for specified module)
make distclean-<module>	(clean everything for specified module)

LXR Ba

- CVS repo browsab
  - http://r “Refere
  - Regene


ROOT Cross-Reference - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location http://root.cern.ch/lxr/

ROOT Cross-Reference

Starting Points:

/ROOT Click here to start browsing at the root of the directory tree.

ChangeLog Click here to browse the ChangeLog file extracted from the CVS repository.

Text Search: [ ] Find
Use this field to search through all the text.

File Name Search: [ ] Find
Use this field to search for files by name.

Identifier Search: [ ] Find
Use this field to find a particular function, variable, etc.

This is a cross referenced display of the ROOT source code. The sources displayed are those that are currently checked in to the mainline of ROOT CVS server; these pages are updated many times a day, so they should be pretty close to the latest-and-greatest.

It's possible to search through the entire ROOT source text; or to search for files whose name matches a pattern; or to search for the definitions of particular functions, variables, etc.

The individual files of the source code are formatted on the fly and presented with clickable identifiers. An *identifier* is a macro, typedef, struct, enum, union, function, function prototype or variable. Clicking on them shows you a summary of how and where they are used.

The free-text search command is implemented using Glimpse, so all the capabilities of Glimpse are available. Regular expression searches are especially useful.

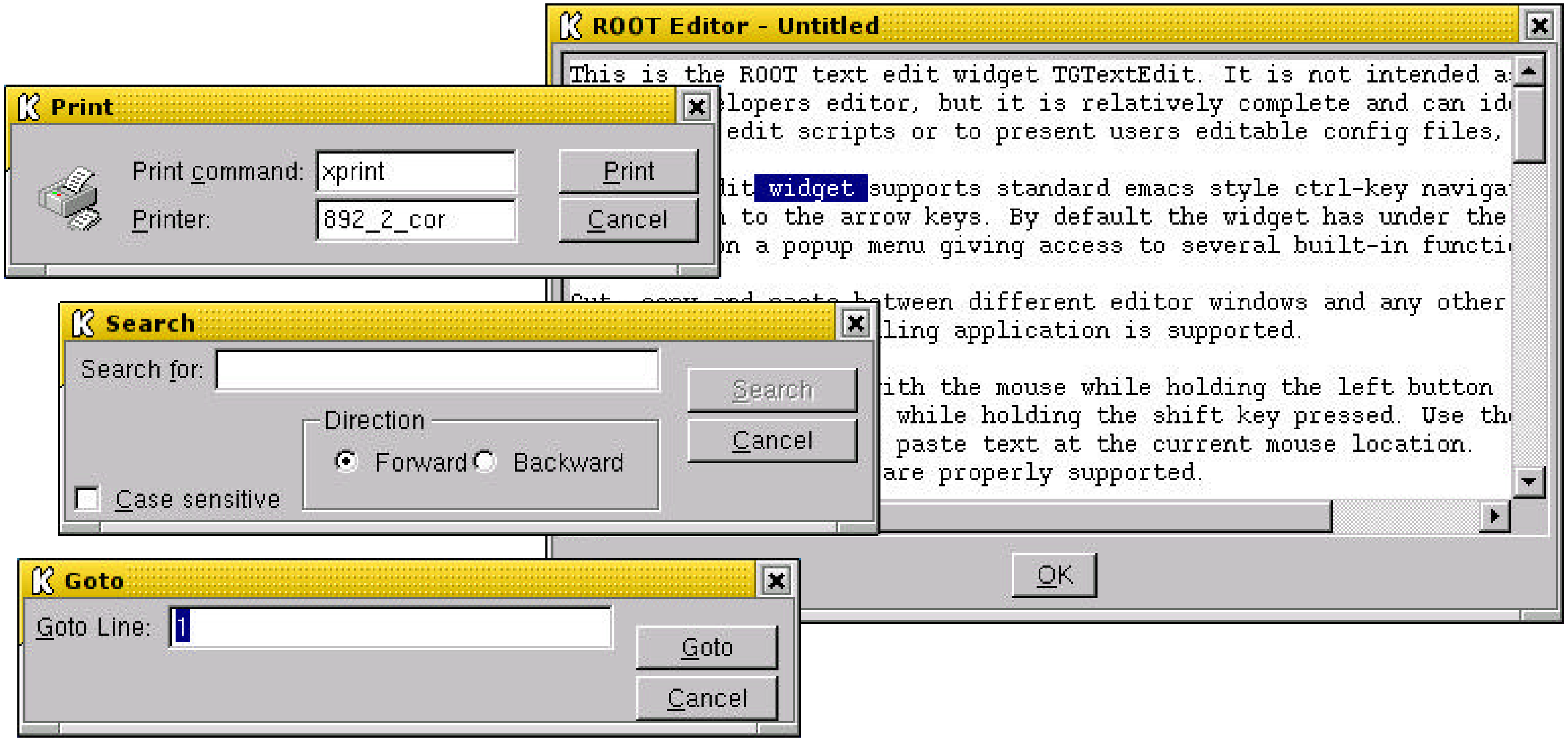
(Don't use use a web-crawler to try and download all of these pages; the CGIs will feed you several *gigabytes* worth of generated HTML!)

The pages here are generated by the LXR tool, which was originally written to display the source code of the Linux kernel (LXR stands for ``Linux Cross Reference.") Check out the main LXR site for more information.

Thanks to Arne Georg Gleditsch and Per Kristian

New ROOT GUI Widgets

- Simple text editor: TGTextEditor



```
This is the ROOT text edit widget TGTextEdit. It is not intended a
lopers editor, but it is relatively complete and can id
edit scripts or to present users editable config files,

lit widget supports standard emacs style ctrl-key naviga
 to the arrow keys. By default the widget has under the
n a popup menu giving access to several built-in functi

Cut, copy and paste between different editor windows and any other
ling application is supported.

ith the mouse while holding the left button
 while holding the shift key pressed. Use th
 paste text at the current mouse location.
are properly supported.
```

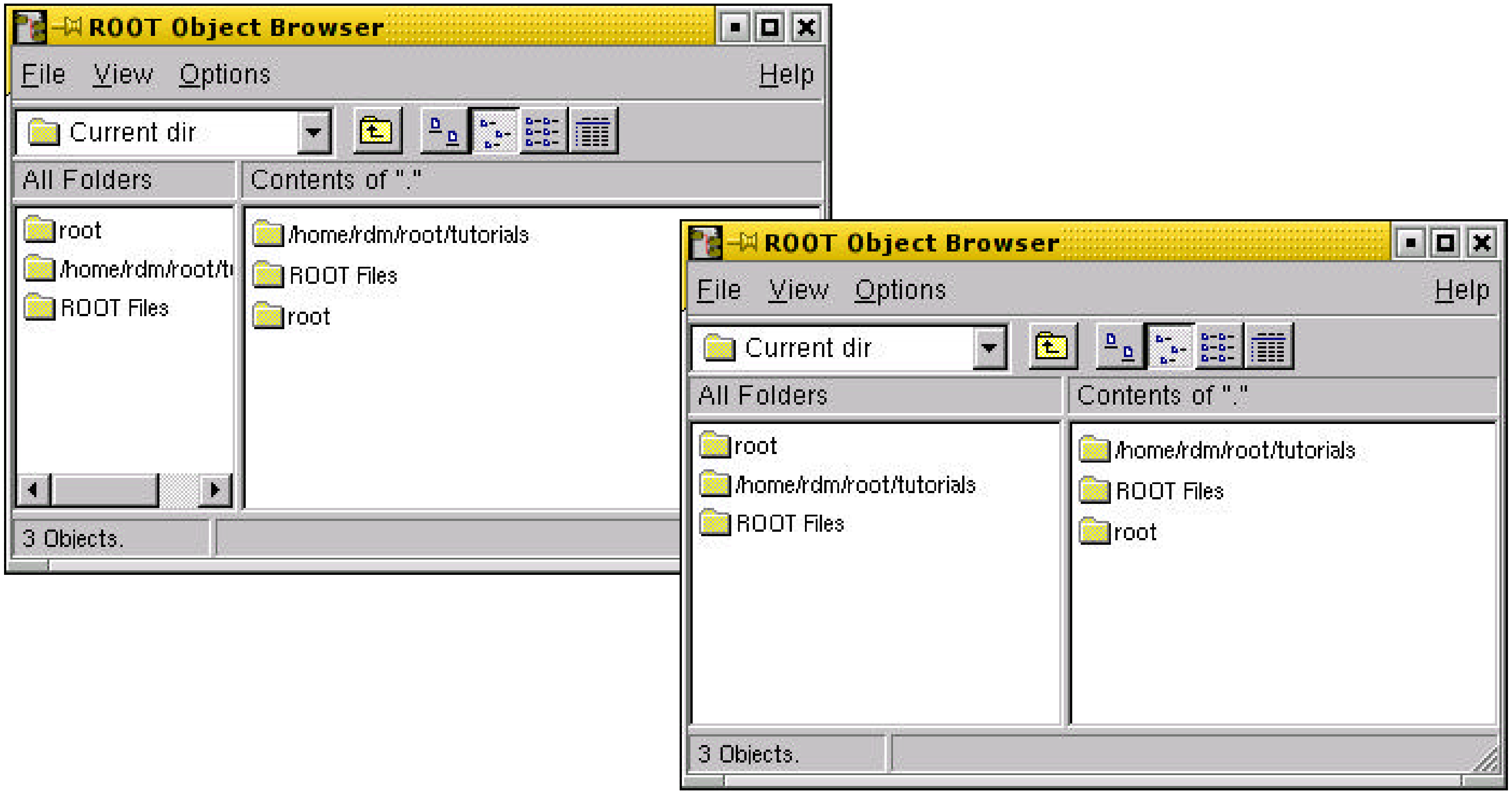
Print command: xprint
Printer: 892_2_cor

Search for:
Direction: (● Forward) ○ Backward
☐ Case sensitive

Goto Line: 1

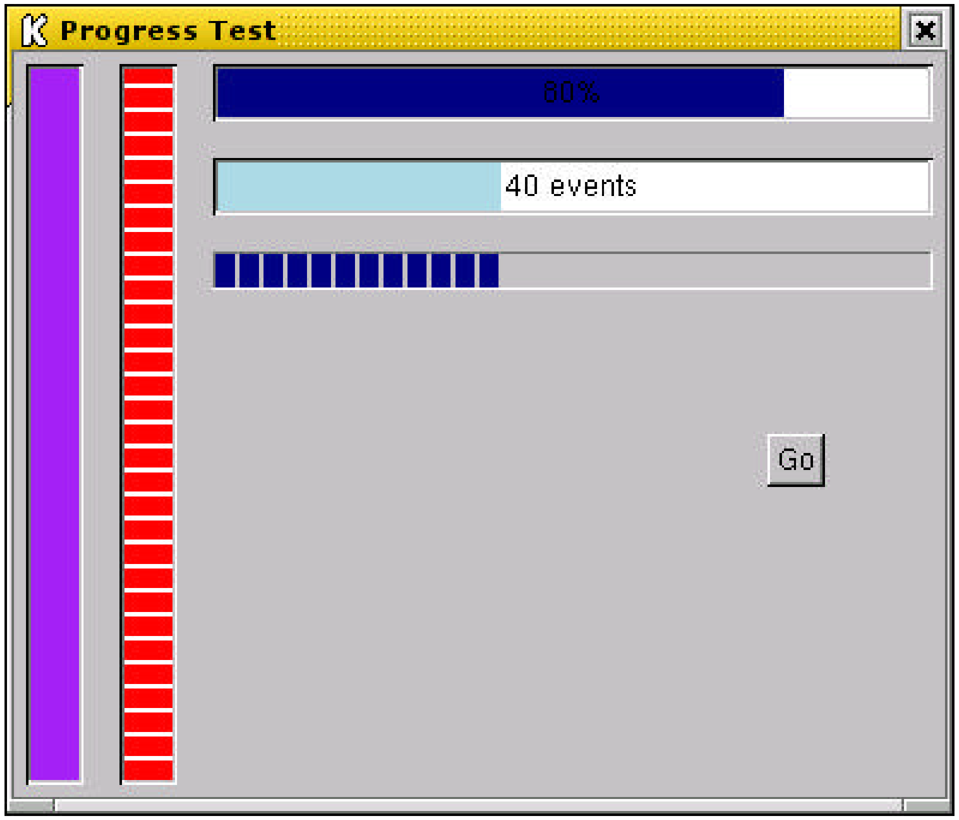
New ROOT GUI Widgets

- Window splitter: TGSplitter



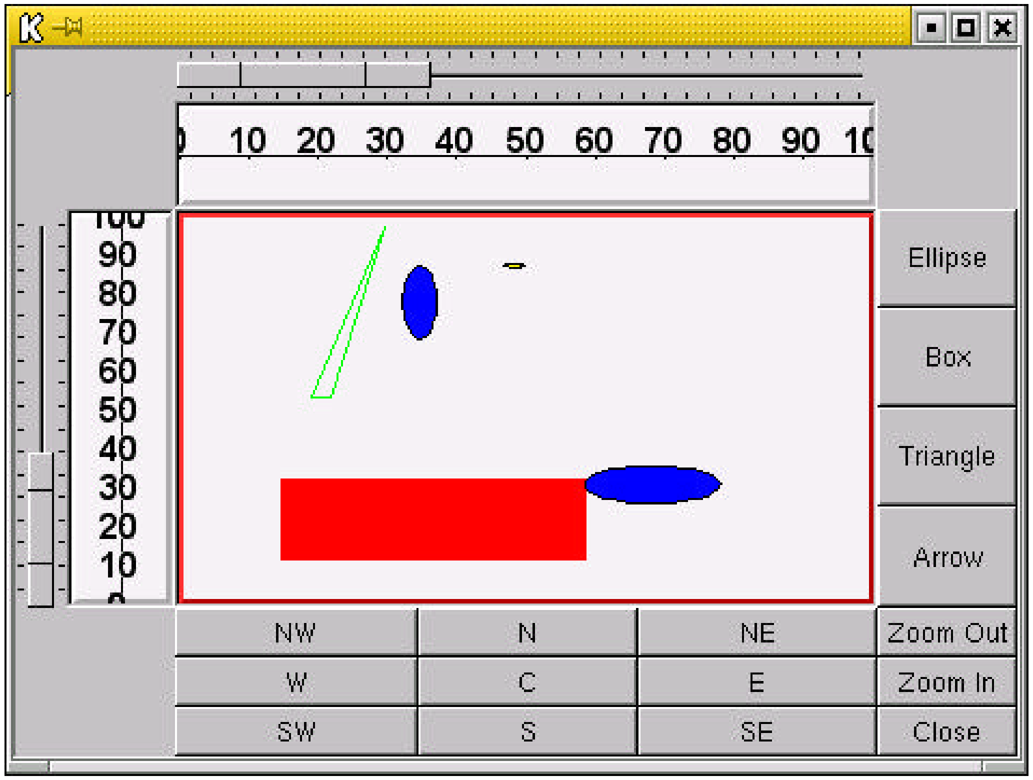
New ROOT GUI Widgets

- Progress bars: TGProgressBar



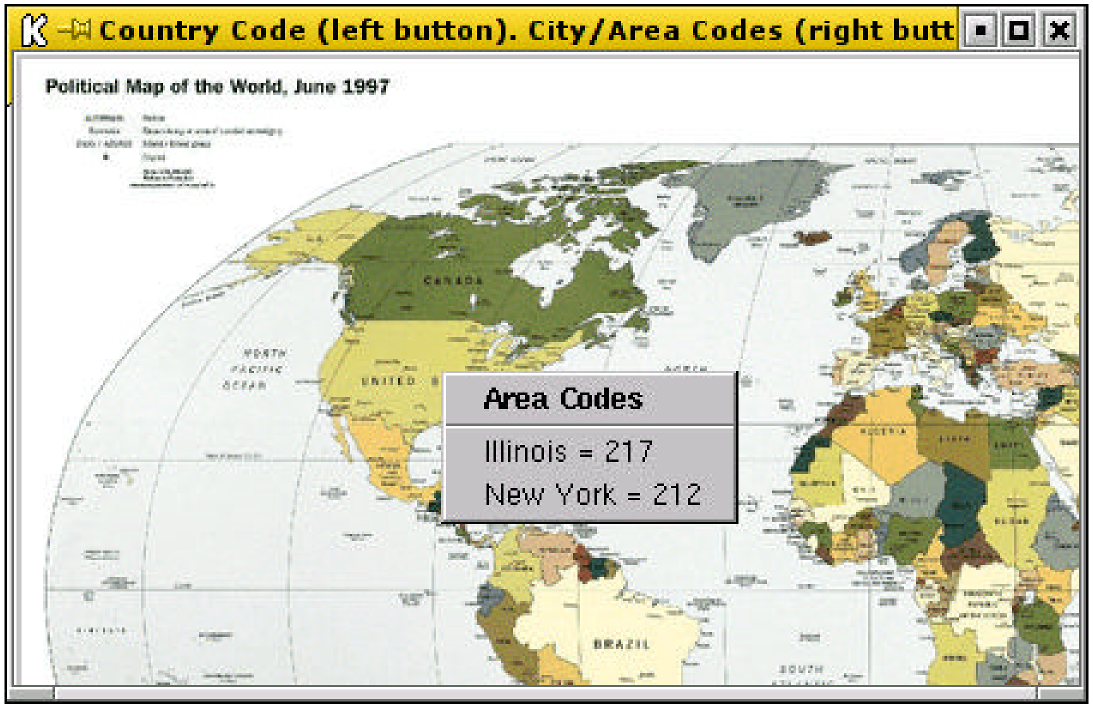
New ROOT GUI Widgets

- Table layout: TGTableLayout
  - By Brett Viren



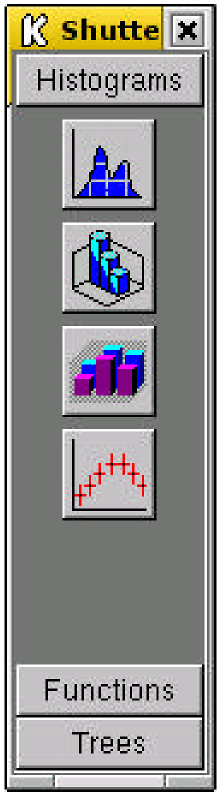
New ROOT GUI Widgets

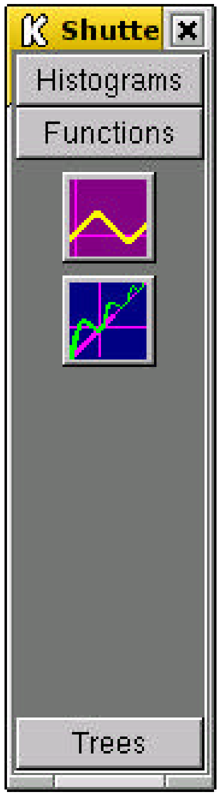
- Image map: TGImageMap
  - By Valeriy Onuchin

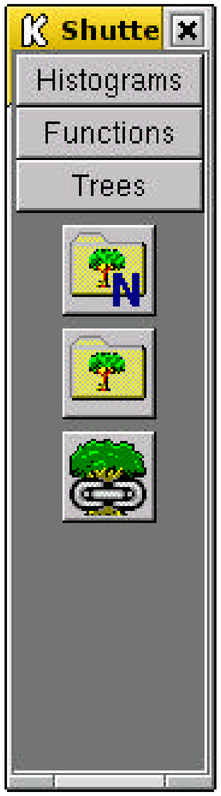


New ROOT GUI Widgets

- Shutter box: TGShutter

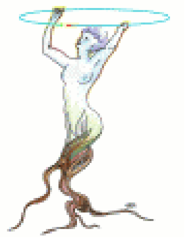






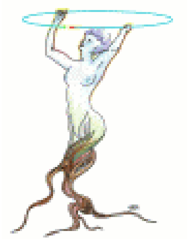
Signals and Slots

by Valeriy Onchin



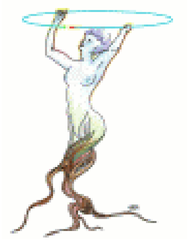
- Integration of signal and slot mechanism into the ROOT core
  - TQObject, TQConnection, TQClass, ...
- Signal and slots were pioneered by Trolltech in their Qt GUI toolkit
- This mechanism facilitates component programming since it allows a total decoupling of the interacting classes

Signals and Slots Example: Emitting a Signal



```
class A {
RQ_OBJECT()
private:
   Int_t  fValue;
public:
   A() { fValue = 0; }
   Int_t  GetValue() const { return fValue; }
   void   SetValue(Int_t);      //*SIGNAL*
};
```

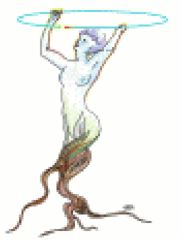
Signals and Slots Example: Emitting a Signal



```
void A::SetValue(Int_t v)
{
   if (v != fValue) {
      fValue = v;
      Emit("SetValue(Int_t)", v);
   }
}
```

```
void TGButton::Clicked()
{
   Emit(“Clicked()");
}
```

Signals and Slots Example: Connecting a Signal to a Slot

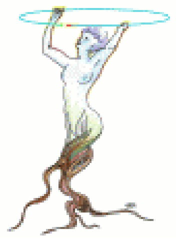


```
A *a = new A();
A *b = new A();
a->Connect("SetValue(Int_t)", "A", b, "SetValue(Int_t)");


a->SetValue(79);
b->GetValue();          // this would is now 79
```

```
fButton->Connect("Clicked()", "MyFrame", this, "DoButton()");
```

Signals and Slots



- The ROOT signal and slot system uses the dictionary information and interpreter to connect signals to slots
- Signals are emitted by:
  - TVirtualPad (TCanvas and TPad)
  - TSysEvtHandler (TTimer, TFileHandler)
  - All GUI widgets
- Let your classes emit signals whenever they change a significant state that others might be interested in

More RDBMS Interfaces

- In addition to original MySQL interface:
- Oracle

  by Michael Dahlinger of GSI
  http://www.gsi.de/computing/root/OracleAccess.htm
- PostgreSQL

  by Gian Paolo Ciceri
- RDBC, a version of JDBC on top of ODBC

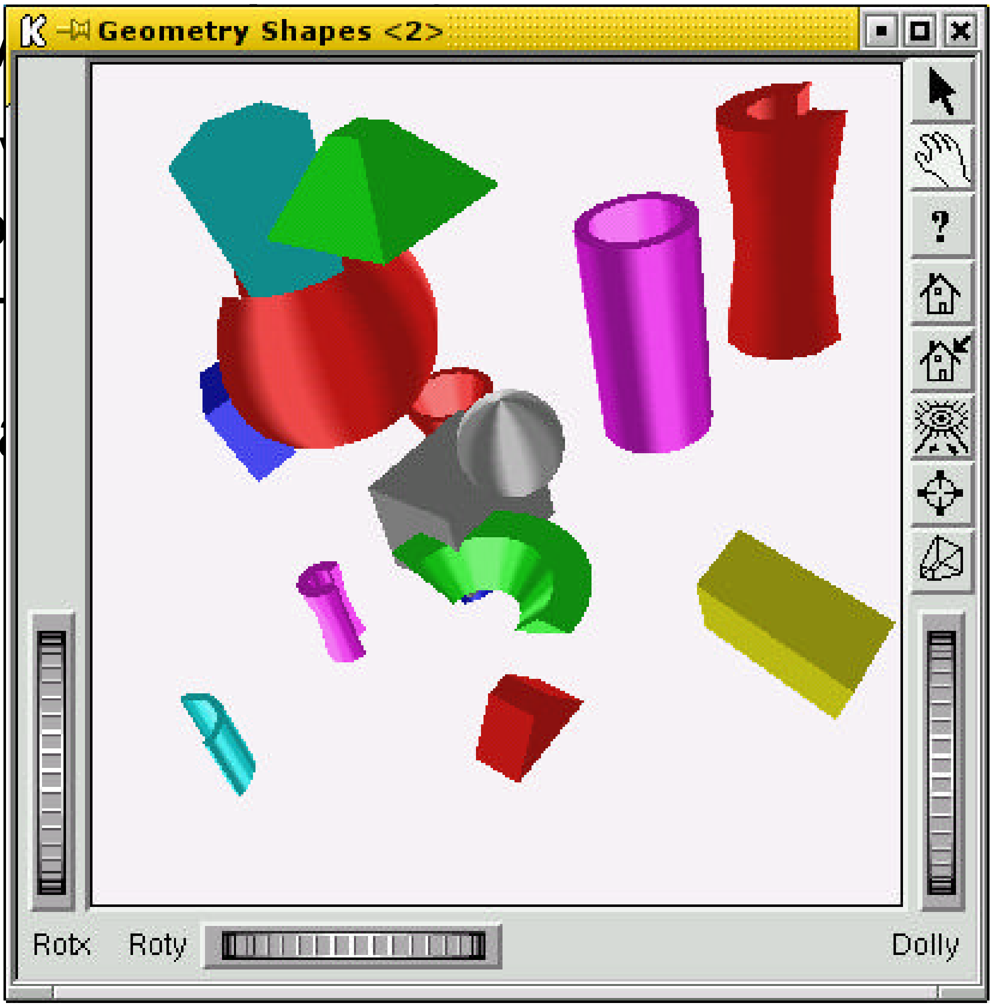
  by Valeriy Onuchin (see talk on Thursday morning)

OpenInventor Integration

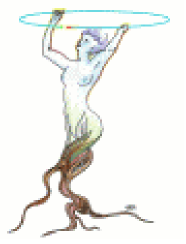
- Basic interface by Valery Fine
- Problems: two event loops, ROOT and Xt
- Merging of Xt in ROOT event loop:
  - dispatch events for Xt managed windows
  - let Xt check and process its timers
- See TRootOIViewer class
- Same technique can be used to embed a Motif based application in ROOT

OpenInventor Interface

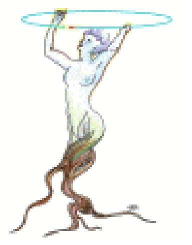
- To get OpenInv
  - Install Open In
    (http://oss.sgi.com/p
  - Re-make ROOT
  - Activate via TC



New Networking Classes



- Parallel socket classes (TPSocket and TPServerSocket)
  - Message striping over multiple sockets
- Parallel ftp class (TFTP)
- Parallel remote file access using parallel version of TNetFile and rootd
- More on these classes in my Friday morning talk



The PROOF System

- Parallel ROOT Facility, goal:
  - parallel execution of scripts
  - parallel analysis of chains of trees
- Further infrastructure developments of the PROOF system
  - config file structure, proof daemons, authentication, client -> master -> slaves communication, etc.
- For more see my Friday morning talk